Distributed Program Environment

Computing Environments

What is Computing Environment?

When we want to solve a problem using a computer, the computer makes use of various devices which work together to solve that problem. There may be a various number of ways to solve a problem. We use the various number of computer devices arranged in different ways to solve different problems. The arrangement of computer devices to solve a problem is said to be a computing environment. The formal definition of the computing environment is as follows.

Computing Environment is a collection of computers which are used to process and exchange information to solve various types of computing problems.

Types of Computing Environments

The following are the various types of computing environments.

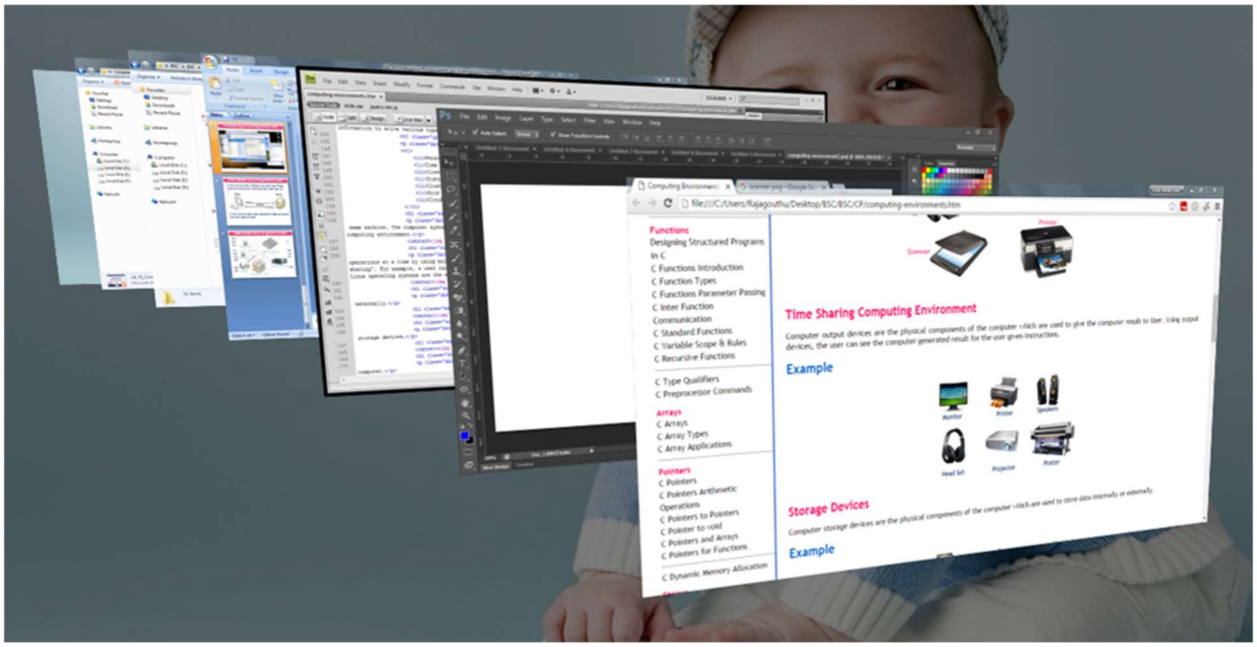
1. Personal Computing Environment
2. Time-Sharing Computing Environment
3. Client-Server Computing Environment
4. Distributed Computing Environment
5. Grid Computing Environment
6. Cluster Computing Environment

Personal Computing Environment

Personal computing is a stand-alone machine. In a personal computing environment, the complete program resides on the stand-alone machine and executed from the same machine. Laptops, mobile devices, printers, scanners and the computer systems we use at home, office are the examples for the personal computing environment.

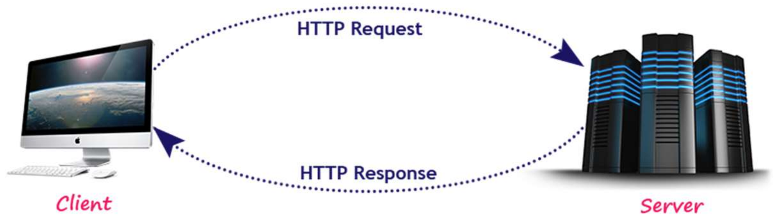
Time-Sharing Computing Environment

The time-sharing computing environment is a stand-alone computer in which a single user can perform multiple operations at a time by using a multitasking operating system. Here the processor time is divided among different tasks and this is called “Time-sharing”. For example, a user can listen to music while writing something in a text editor. Windows 95 and later versions of Windows OS, iOS and Linux operating systems are the examples for this computing environment.



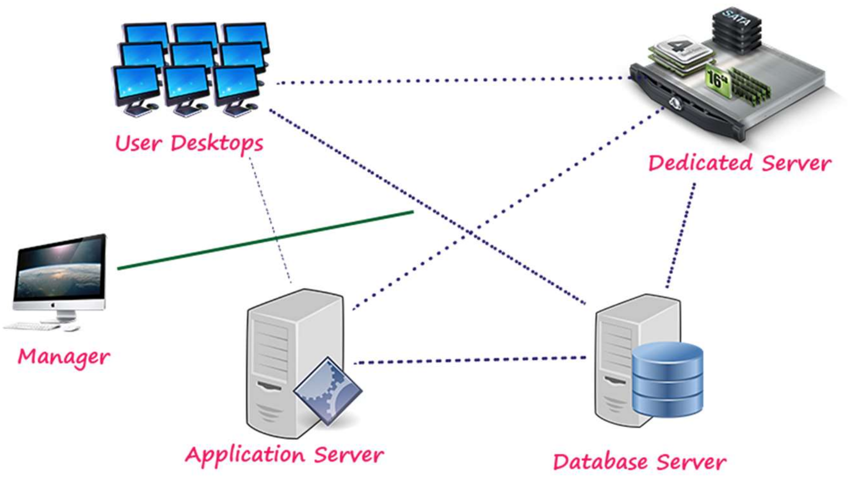
Client-Server Computing Environment

The client-server environment contains two machines (Client machine and Server machine). These both machines will exchange the information through an application. Here Client is a normal computer like PC, Tablet, Mobile, etc., and Server is a powerful computer which stores huge data and manages the huge amount of file and emails, etc., In this environment, client requests for data and server provides data to the client. In the client-server environment, the communication between client and server is performed using HTTP (Hyper Text Transfer Protocol).



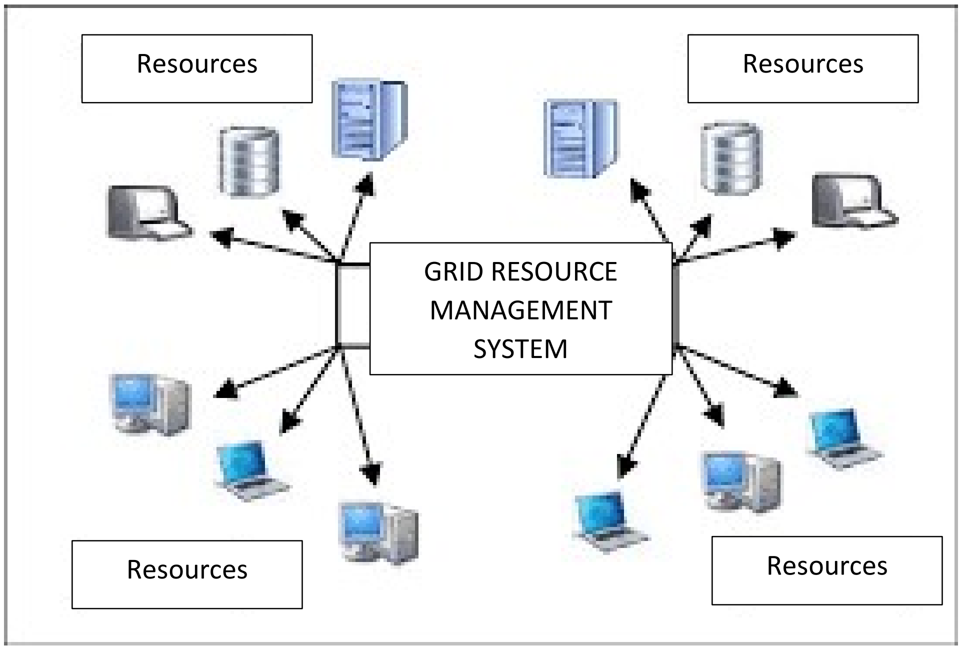
Distributed Computing Environment

In the distributed computing environment, the complete functionality of the software is not on a single computer but is distributed among multiple computers. Here we use a method of computer processing in which different programs of an application run simultaneously on two or more computers. These computers communicate with each other over a network to perform the complete task. In a distributed computing environment, the data is distributed among different systems and that data is logically related to each other.



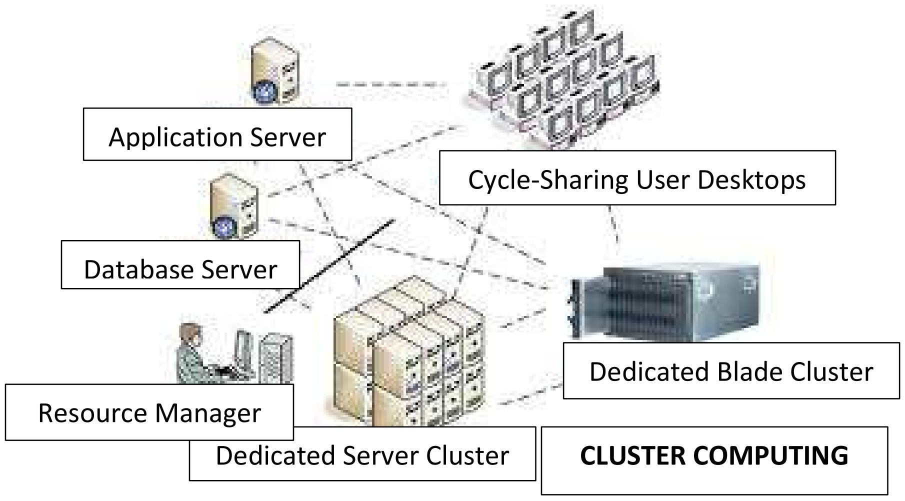
Grid Computing Environment

Grid computing is a collection of computers from different locations. All these computers work for a common problem. A grid can be described as a distributed collection of a large number of computers working for a single application.



Cluster Computing Environment

Cluster computing is a collection of interconnected computers. These computers work together to solve a single problem. In a cluster computing environment, a collection of systems work together as a single system.



Computer Languages

What is Computer Language?

Generally, we use human languages, like English or emoji, to make communication between two persons. That means when we want to make communication between two persons, we need a language through which persons can express their feelings. Similarly, when we want to make communication between user and computer or between two or more computers, we need a language through which user can give information to the computer and vice versa. When a user wants to give any instruction to the computer the user needs a specific language and that language is known as a ***computer language***.

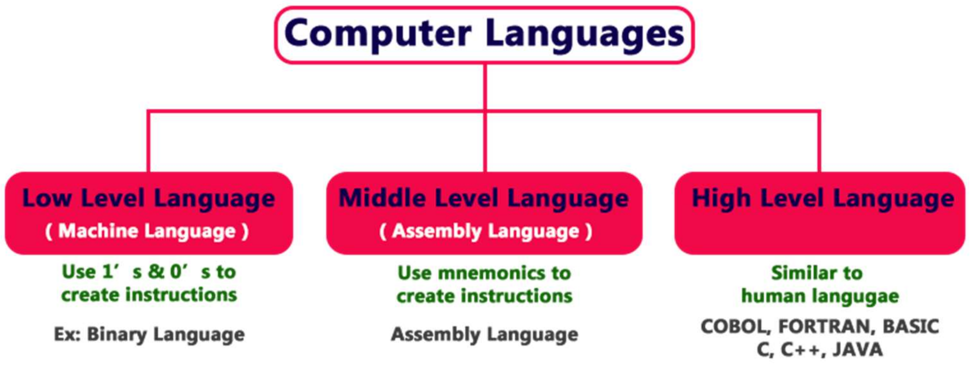
The user interacts with the computer using programs and that programs are created using computer programming languages like **C**, C++, Java, etc.,

Computer languages are the languages through which the user can communicate with the computer by writing program instructions.

Every computer programming language contains a set of predefined words and a set of rules (syntax) that are used to create instructions of a program. Syntax, semantics, and pragmatics. Pragmatics is a subfield of linguistics and signage-symbols for the ways in which context contributes to meaning.

Computer Languages Classification

Over the years, computer languages have been evolved from Low-Level to High-Level Languages. In the earliest days of computers, only Binary Language was used to write programs. The computer languages are classified as follows...



Low-Level Language (Machine Language)

Low-Level language is the only language which can be understood by the computer.

Binary Language is an example of a low-level language. Low-level language is also known as **Machine Language**. The binary language contains only two symbols 1 & 0. All the instructions of binary language are written in the form of binary numbers 1's & 0's. A computer can directly understand the binary language. Machine language is also known as the **Machine Code**.

As the CPU directly understands the binary language instructions, it does not require any translator. CPU directly starts executing the binary language instructions and takes very less time to execute the instructions as it does not require any translation. Low-level language is considered as the First Generation Language (1GL).

Advantages

- A computer can easily understand the low-level language.
- Low-level language instructions are executed directly without any translation.
- Low-level language instructions require very less time for their execution.

Disadvantages

- Low-level language instructions are very difficult to use and understand.
- Low-level language instructions are machine-dependent, that means a program written for a particular machine does not execute on another machine.
- In low-level language, there is more chance for errors and it is very difficult to find errors, debug and modify.

Middle-Level Language (Assembly Language)

Middle-level language is a computer language in which the instructions are created using symbols such as letters, digits and special characters.

Assembly language is an example of middle-level language. In assembly language, we use predefined words called **mnemonics**. Binary code instructions in low-level language are replaced with mnemonics and operands in middle-level language. But the computer cannot understand mnemonics, so we use a translator called **Assembler** to translate mnemonics into binary language. Assembler is a translator which takes assembly code as input and produces machine code as output. That means, the computer cannot understand middle-level language, so it needs to be translated into a low-level language to make it understandable by the computer. Assembler is used to translate middle-level language into low-level language.

Advantages

- Writing instructions in a middle-level language is easier than writing instructions in a low-level language.
- Middle-level language is more readable compared to low-level language.
- Easy to understand, find errors and modify.

Disadvantages

- Middle-level language is specific to a particular machine architecture, that means it is machine-dependent.
- Middle-level language needs to be translated into low-level language.
- Middle-level language executes slower compared to low-level language.

High-Level Language

A high-level language is a computer language which can be understood by the users. **The high-level language is very similar to human languages and has a set of grammar rules that are used to make instructions more easily.** Every high-level language has a set of predefined words known as Keywords and a set of rules known as Syntax to create instructions. **The high-level language is easier to understand for the users but the computer cannot understand it.** High-level language needs to be converted into the low-level language to make it understandable by the computer. We use **Compiler** or **interpreter** to convert high-level language to low-level language.

Languages like COBOL, FORTRAN, BASIC, C, C++, JAVA, etc., are examples of high-level languages. All these programming languages use human-understandable language like English to write program instructions. These instructions are converted to low-level language by the compiler so that it can be understood by the computer.

Advantages

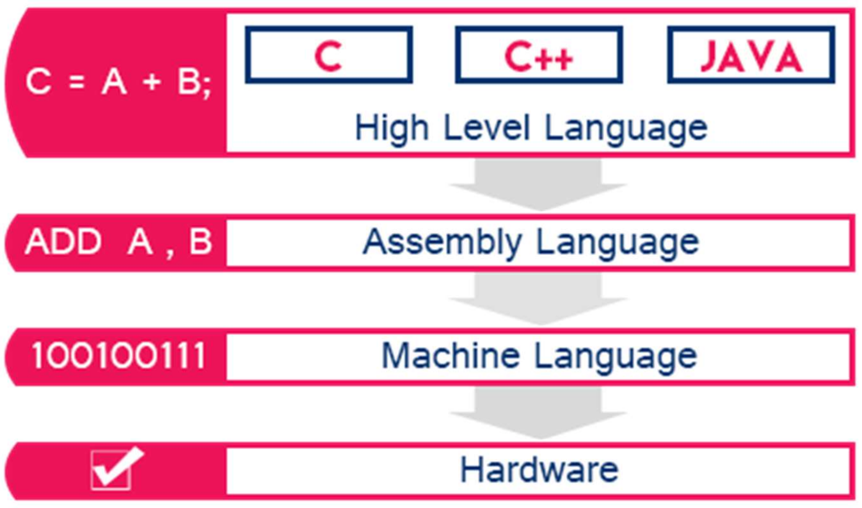
- Writing instructions in a high-level language is easier.
- A high-level language is more readable and understandable.
- The programs created using high-level language runs on different machines with little change or no change.
- Easy to understand, create programs, find errors and modify.

Disadvantages

- High-level language needs to be translated into low-level language.
- High-level language executes slower compared to middle and low-level languages.

Understanding Computer Languages

The following figure provides a few key points related to computer languages.



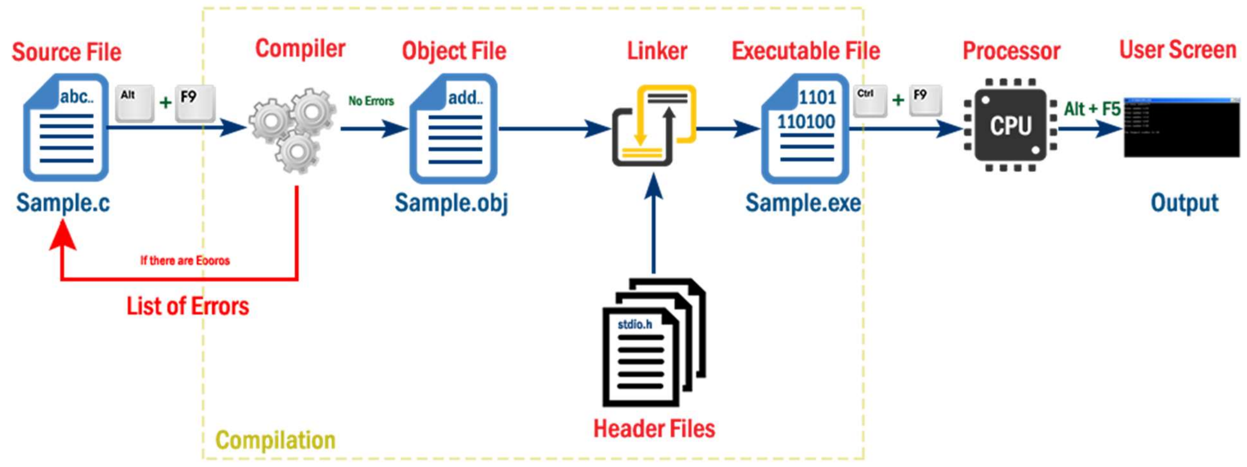
From the above figure, we can observe the following key points...

- The programming languages like C, C++, Java, etc., are written in High-level language which is more comfortable for the developers.
- A high-level language is closer to the users.
- Low-level language is closer to the computer. Computer hardware can understand only the low-level language (Machine Language).
- The program written in the high-level language needs to be converted to low-level language to make communication between the user and the computer.
- Middle-level language is not closer to both user and computer. We can consider it as a combination of both high-level language and low-level language.

Creating and Running C Program

Execution Process of a C Program

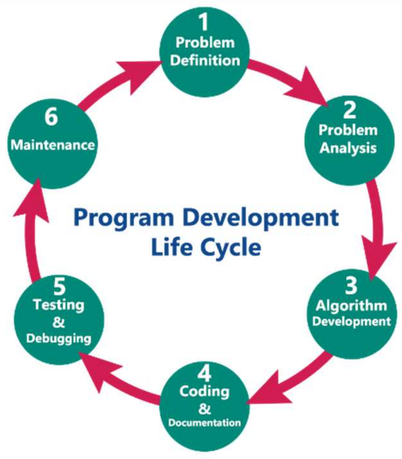
When we execute a C program it undergoes with the following process…



The file which contains c program instructions in a high-level language is said to be source code. Every c program source file is saved with .c extension, for example, the program named **Sample.c**.

Program Development Life Cycle

When we want to develop a program using any programming language, we follow a sequence of steps. These steps are called phases in program development. The program development life cycle is a set of steps or phases that are used to develop a program in any programming language. Generally, the program development life cycle contains 6 phases, they are as follows….

1. Problem Definition **2.** Problem Analysis **3.** Algorithm Development **4.** Coding & Documentation **5.** Testing & Debugging **6.** Maintenance	 1 Problem Definition 2 Problem Analysis 3 Algorithm Development 4 Coding & Documentation 5 Testing & Debugging 6 Maintenance Program Development Life Cycle

1. Problem Definition

In this phase, we define the problem statement and we decide the boundaries of the problem. In this phase we need to understand the problem statement, what is our requirement, what should be the output of the problem solution. These are defined in this first phase of the program development life cycle.

2. Problem Analysis

In phase 2, we determine the requirements like variables, functions, etc. to solve the problem. That means we gather the required resources to solve the problem defined in the problem definition phase. We also determine the bounds of the solution.

3. Algorithm Development

During this phase, we develop a step by step procedure to solve the problem using the specification given in the previous phase. This phase is very important for program development. That means we write the solution in step by step statements.

4. Coding & Documentation

This phase uses a programming language to write or implement the actual programming instructions for the steps defined in the previous phase. In this phase, we construct the actual program. That means we write the program to solve the given problem using programming languages like C, C++, Java, etc.,

5. Testing & Debugging

During this phase, we check whether the code written in the previous step is solving the specified problem or not. That means we test the program whether it is solving the problem for various input data values or not. We also test whether it is providing the desired output or not.

6. Maintenance

During this phase, the program is actively used by the users. If any enhancements found in this phase, all the phases are to be repeated to make the enhancements. That means in this phase, the solution (program) is used by the end-user. If the user encounters any problem or wants any enhancement, then we need to repeat all the phases from the starting, so that the encountered problem is solved or enhancement is added.

C Background

C is a structured programming language. It is also known as function orientated programming language. C programming language was developed in the year of **1972** by **Dennis Ritchie** at Bell Laboratories in the USA (AT&T).

In the year of 1968, research was started by Dennis Ritchie on programming languages like BCPL, CPL. The main aim of his research was to develop a new language to create an OS called UNIX. After four years of research, a new programming language was created with solutions for drawbacks in languages like BCPL & CPL. In the year of 1972, the new language was introduced with the name “**Traditional C**”.

The name 'c' was selected from the sequence of previous language ‘B’ (BCPL) because most of the features of 'c' were derived from BCPL (B language).

The first outcome of the c language was the UNIX operating system. The initial UNIX OS was completely developed using 'c' programming language.

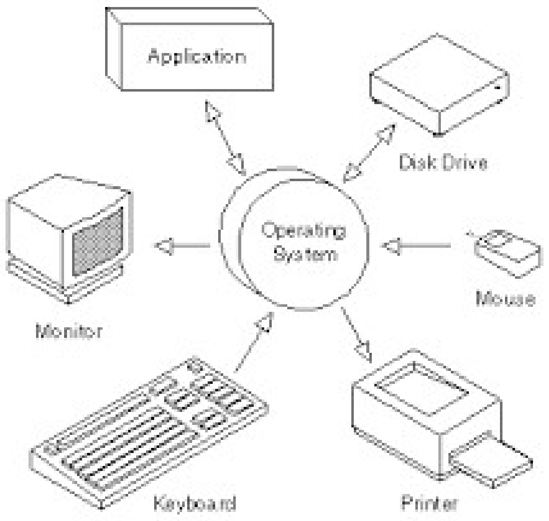
The founder of the ‘C’ language, **Dennis Ritchie is known as “Father of C”** and also **“Father of UNIX”**.

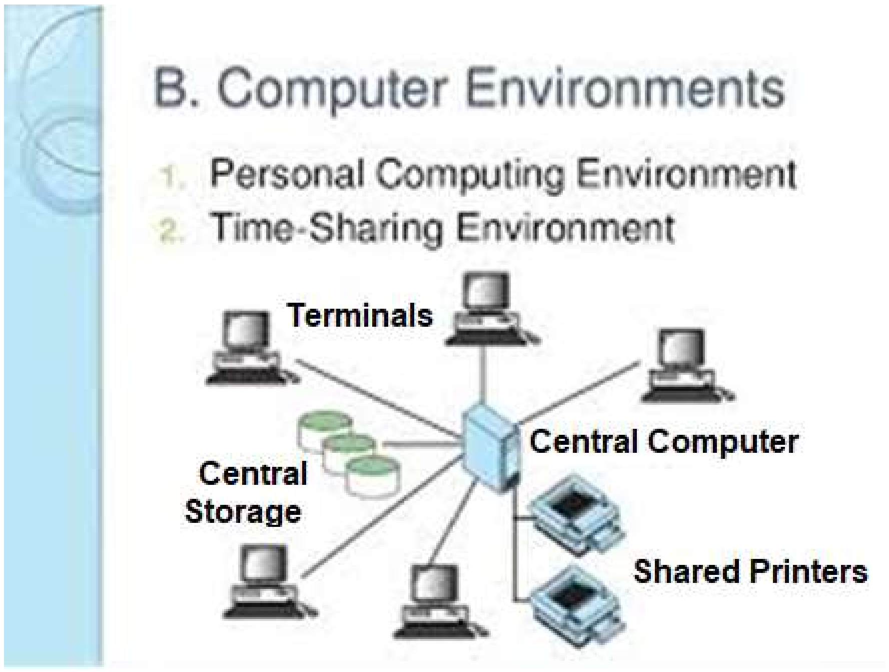
The c programming language is very popular because it is reliable, simple and easy to use and it is the base for almost all the other programming languages.

SINGLE TASKING

MULTI-TASKING

SIMULTANEOUS TASKING DEVICES

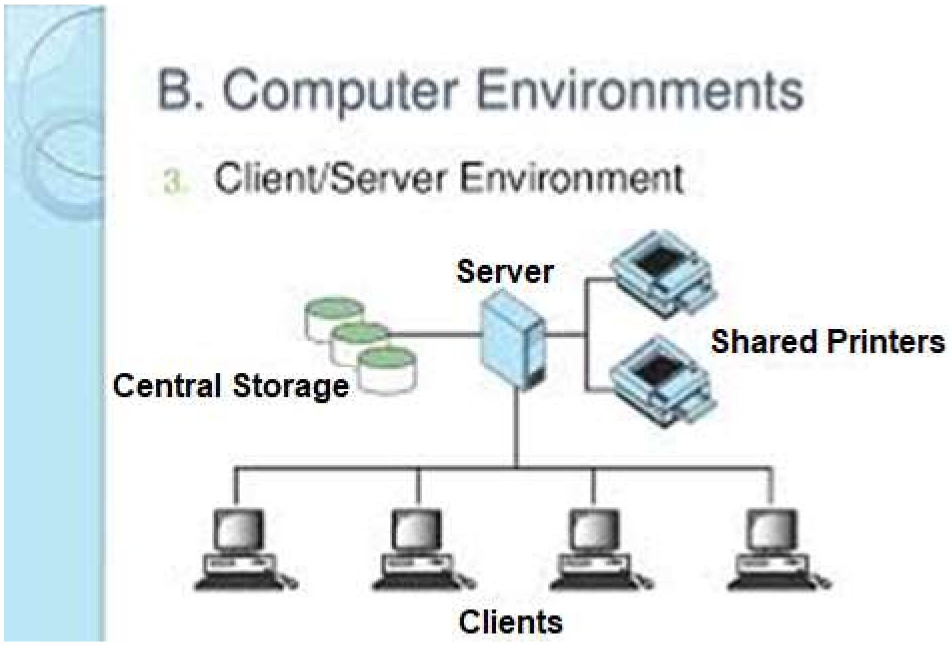




B. Computer Environments

1. Personal Computing Environment
2. Time-Sharing Environment

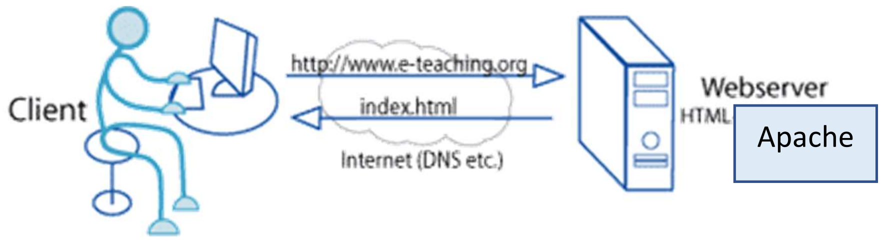
1. Personal Computing Environment

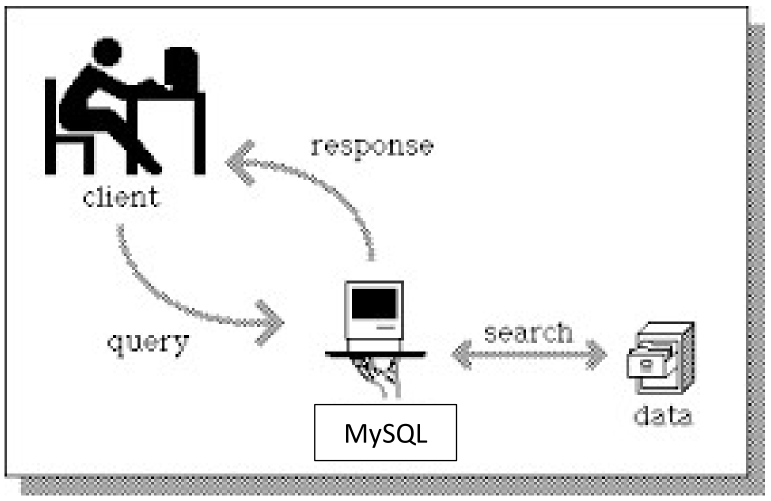


B. Computer Environments

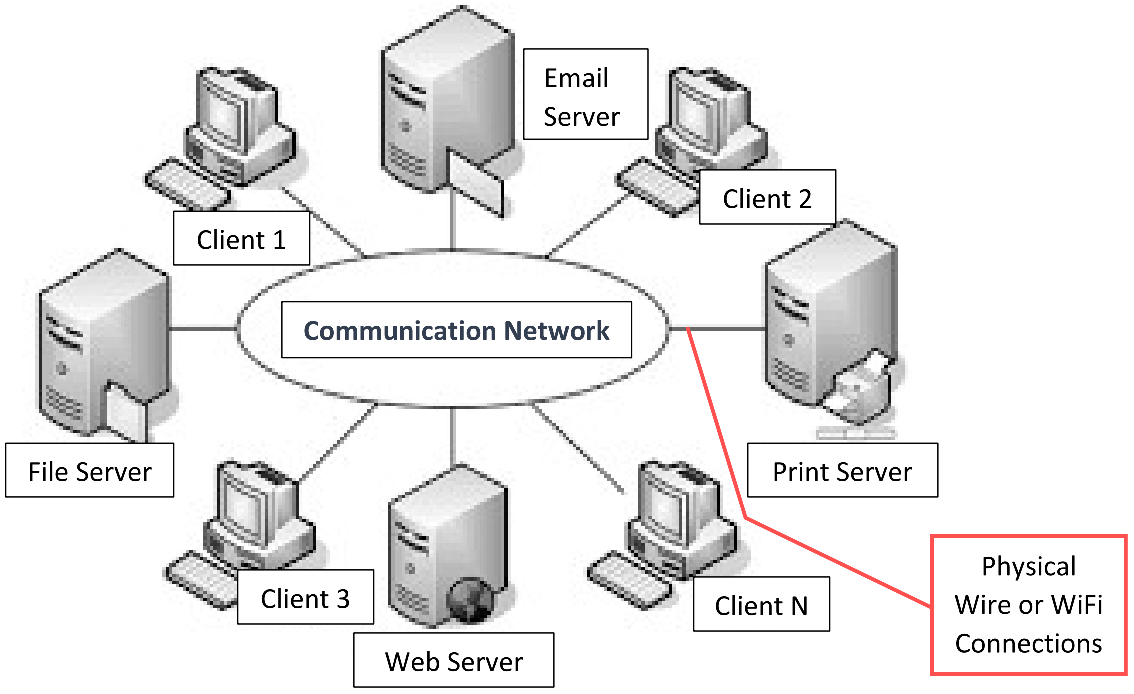
3. Client/Server Environment

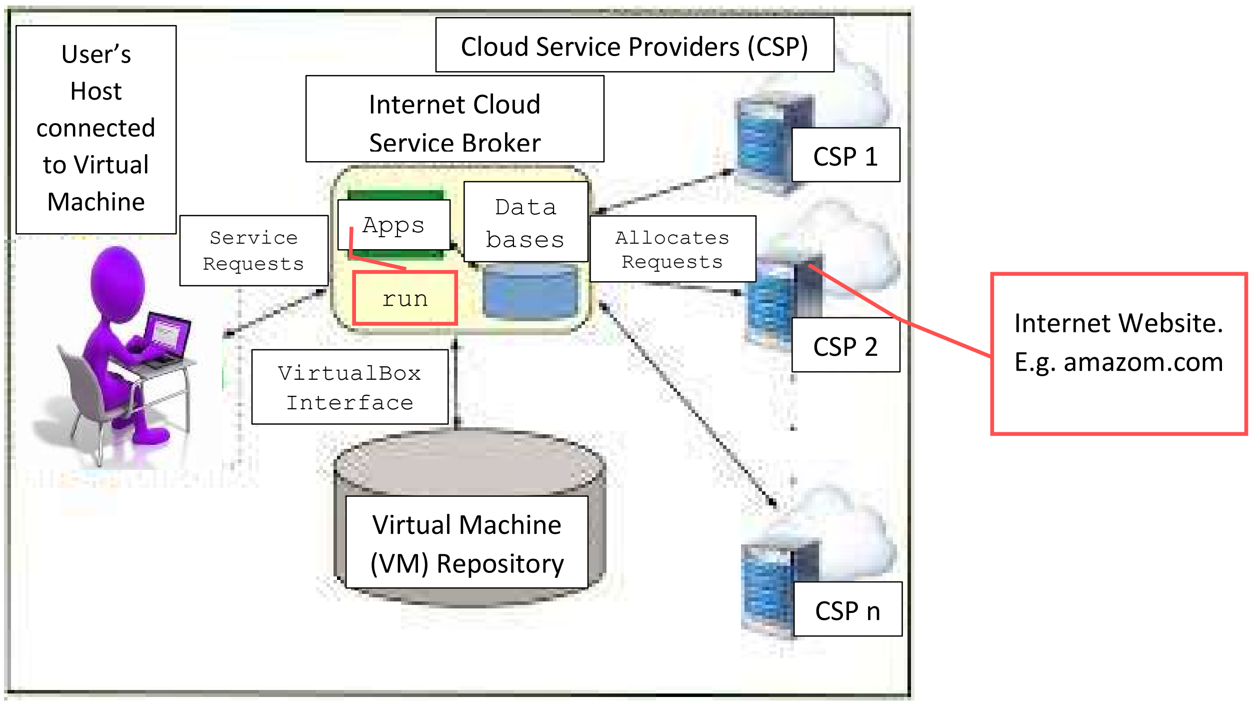
Client Cloud Computing Environment





Device Connection Physically Wired/WiFi to Network

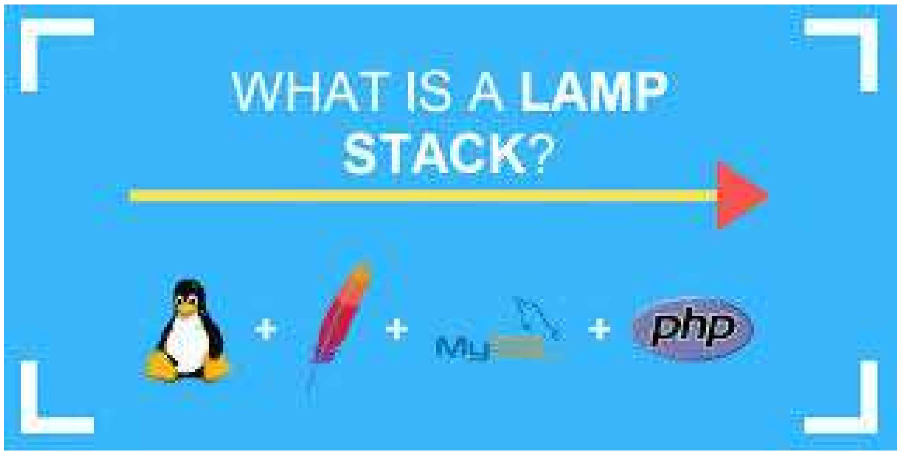



User's Host connected to Virtual Machine
Cloud Service Providers (CSP)
Internet Cloud Service Broker
CSP 1
Apps
Data bases
Service Requests
Allocates Requests
run
CSP 2
Internet Website. E.g. amazom.com
VirtualBox Interface
Virtual Machine (VM) Repository
CSP n

VIRTUAL MACHINE

Emulating Linux Operating System Environment

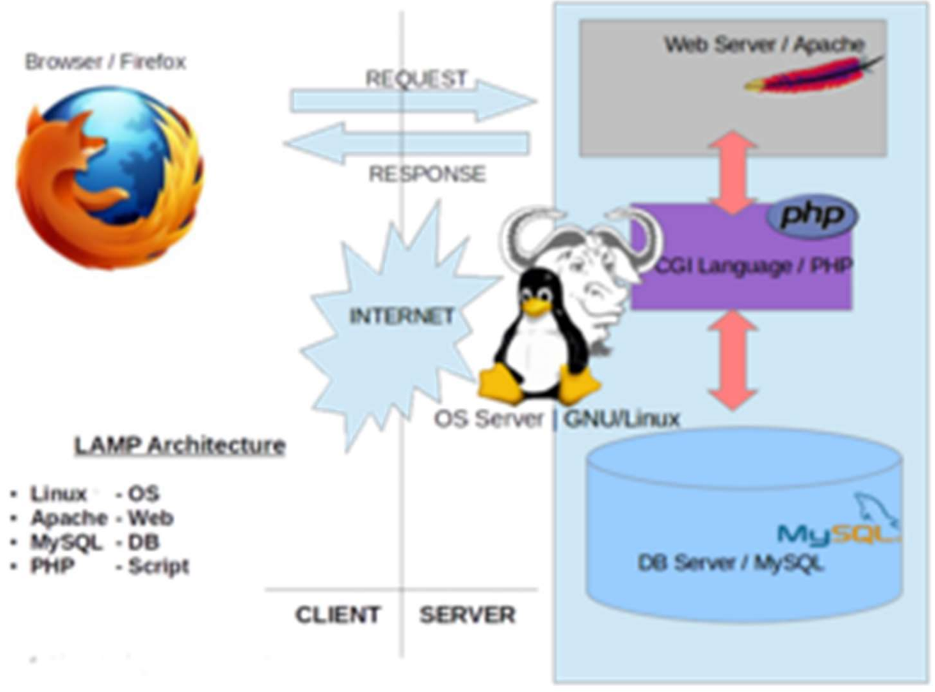
Linux Apache MySQL phpMyAdmin



LAMP Stack

Linux Apache MySQL PHP

LAMP ARCHITECTURE PROCESSES


Browser / Firefox
REQUEST
RESPONSE
INTERNET
Web Server / Apache
php
CGI Language / PHP
OS Server | GNU/Linux
MySQL
DB Server / MySQL
LAMP Architecture
• Linux - OS
• Apache - Web
• MySQL - DB
• PHP - Script
CLIENT
SERVER